

Text Segmentation with Combination of Text Clustering by String Vector based KNN and AHC Algorithm

Taeho Jo
President
Alpha AI Research
Cheongju, South Korea
tjo018@naver.com

Abstract—In this research, we propose the string vector based KNN algorithm and the string vector based AHC algorithm for automating the text segmentation by combining itself with the text clustering. In the previous work, even if the KNN version was proposed as an approach to the text segmentation, a domain should be mentioned, in addition to an input text. In this research, text clusters are made based on their contents by the text clustering, and adjacent paragraph pairs from an input text are classified into boundary or continuance by a classifier which corresponds to its most similar cluster. The string vector based AHC algorithm, and the string vector based KNN algorithm are adopted respectively for the text clustering and the text segmentation. The goals of this research are to automate the text segmentation and to improve the performance of both tasks, using the two string vector-based versions.

I. INTRODUCTION

The text segmentation is defined as the process of segmenting a text logically into subtexts. It is interpreted into the classification of adjacent paragraph pairs into boundary or continuance, to judge whether a boundary is put between paragraphs, or not. The process of text segmentation is to partition a text into paragraphs and classify each adjacent paragraph pair into one of the two categories. A boundary is put between paragraphs in each pair which is classified into boundary. A text is segmented into subtexts by the boundaries, and the subtexts are treated as independent texts.

This research is motivated by the requirement of defining domains subjectively for designing the text segmentation system. The text segmentation is mapped into a binary classification of paragraph pairs for each domain; the text segmentation system is composed with binary classification systems as many as domains. The process of designing the text segmentation system is to predefine domains manually, gather sample paragraph pairs for each domain, and allocate a binary classifier domain by domain. One more motivation for this research is the validation of excellent performances of the string vector based KNN algorithm and the string vector based AHC algorithm in the text segmentation and the text clustering. In this research, we connect the text segmentation with the text clustering and adopt the string vector-based machine learning algorithms for both tasks.

In this research, we propose that the text segmentation should relate to the text clustering, and the string vector based KNN algorithm and the string vector based AHC algorithm are applied to both tasks. In this research, we initially prepare the text collection where each text is segmented into subtexts by the topic transition. The semantic similarity between two string vectors is defined, and the KNN algorithm and the AHC algorithm are modified based on the similarity metric into the string vector-based versions. The collection is segmented into clusters by the text clustering, and in each cluster, which indicates a domain, we gather paragraph pairs which are labeled with one of the two categories and train the binary classifier which performs the text segmentation. A domain is automatically decided to an input text by its similarities with the cluster prototypes for selecting a corresponding text segmentation system.

Let us mention some benefits from this research. The main problems in encoding texts into numerical vectors are solved by encoding them into string vectors. The performances in both tasks which relate to each other are improved by adopting the string vector-based versions of the KNN algorithm and the AHC algorithm. No requirement of defining the domains depending on prior knowledge and subjective views is caused by automating the domain predefinition by the text clustering. No requirement of tagging an input text with its domain manually is caused by automating the process of assigning a domain to an input text based on its similarities with the cluster prototypes.

Let us mention the organization of this paper. In Section II, we explore the previous works which are relevant to this study. In Section III, we describe in detail what is proposed in this research. In Section IV, we mention the significances of this research and the remaining tasks. This paper is composed of the four sections.

II. PREVIOUS WORKS

This section is concerned with previous works which are relevant to this research. It is intended to connect text segmentation with the text clustering for defining domains in a text collection. The direction is to review cases of applying the proposed version of KNN algorithm to the text clustering

and the text segmentation and modifying the KNN algorithm and the AHC algorithm into string vector-based versions for improving their performances. The difference of this research is that the text segmentation is connected to the text clustering for implementing a complete text segmentation system. This section is intended to explore previous works for providing background for this research.

Let us explore the previous works which are concerned with the application of the modified version of the standard AHC algorithm to text clustering. In 2017, Jo initially proposed that the string vector-based version should be applied to the text clustering [4]. In 2019, Jo validated empirically its performances in the text clustering [9]. In 2020, he published the journal article which describes the string vector-based AHC algorithm and its application to the text clustering for its publication [8]. The text clustering system which was implemented in the above works will be connected to the proposed system which is implemented in this research.

Let us explore the previous works on applying the modified version of KNN algorithm to the text segmentation. In 2017, it was initially proposed that the string vector-based version of KNN algorithm should be applied to the text segmentation as a position paper by Jo [5]. In 2019, the modified KNN algorithm was validated in the task by Jo [10]. The journal article which describes in detail the modified KNN algorithm and its application to the text segmentation is finally prepared for its publication by Jo, as well as text segmentation [11]. It was pointed out that the text segmentation is a domain dependent classification, and it is necessary to define domains automatically from a particular text collection.

Let us explore the previous works on the neural networks, NTC (Neural Text Categorizer), where encoding a text into a string vector was initially proposed. In 2010, it was initially invented as an approach to the text classification by Jo [1]. In 2015, it was applied to the topic identification of Arabic texts by Abainia [3]. In 2018, it was evaluated as the most practical neural networks by Lakshmi and Rao [7]. In 2010, the NTSO (Neural Text Self Organizer) was also invented as an approach to the text clustering by Jo [2].

Let us mention the differences of this research from the previous works which are explored above. The text clustering which was implemented in [8] is applied to domain definition for text segmentation. In this research, the domains of text collection are defined by introducing text clustering for upgrading the text segmentation system. Machine learning algorithms will be innovated by inventing various ones which process directly string vectors as their input. We will consider introducing text segmentation to text clustering as the reverse case of this research in the next one.

III. PROPOSED WORKS

This section is concerned with what is proposed in this research. In Section III-A, we describe the process of encoding a text into a string vector and the proposed similarity metric. In Section III-B, we describe the string vector based version of KNN algorithm. In Section III-C, we present the system architecture of the text segmentation system. In Section III-D, we present the system architecture of its extended version by connecting it with the text clustering system.

A. Encoding Proccerss

This section is concerned with the string vector as the text representation, and the similarity between string vectors. A string vector is a finite ordered set of strings; numerical values are replaced by strings as the elements in a vector. It is required to define the similarity matrix for performing the operation on string vectors. The similarity between string vectors is the average over one-to-one similarities between their elements. This section is intended to describe the process of encoding a text into a string and the process of computing the similarity between string vectors.

The process of encoding a text into a string vector is illustrated in Figure 1. A text is indexed into a list of words and their information. The features of a string vector are defined as symbolic forms manually in the feature definition. A string which represents a text is filled with the words which corresponds to the features in the feature value assignment. Refer to [8] for studying the encoding process in more detail.

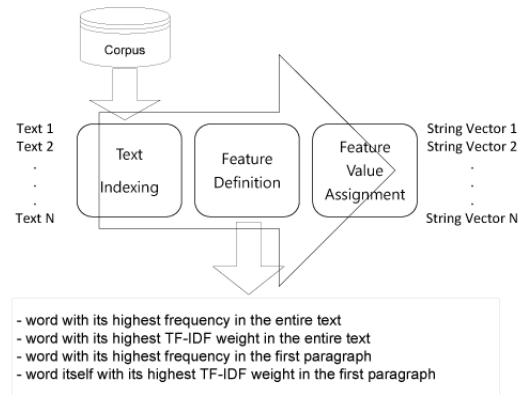


Figure 1. Process of Encoding Texts into String Vectors

The similarity matrix which is the basis for computing the similarity between two string vectors is illustrated in Figure 2. Each row and each column correspond to their own word, and each element in the similarity matrix is the similarity between two words, t_1 and t_2 , as shown in equation (1),

$$s_{ij} = \text{sim}(t_i, t_j) \quad (1)$$

The similarity between two words, t_i and t_j is computed by equation (2),

$$sim(d_i, d_j) = \frac{2\#(t_i \wedge t_j)}{\#(t_i) + \#(t_j)} \quad (2)$$

where $\#(t_i \wedge t_j)$ is the number of texts which include both t_i and t_j , $\#(t_i)$ is the number of texts which includes t_i , and $\#(t_j)$ is the number of texts which includes t_j . The value of similarity between two words, t_i and t_j , is always given as a normalized value between zero and one, as shown in equation (3),

$$0 \leq sim(t_i, t_j) \leq 1 \quad (3)$$

The similarity matrix is used for computing the similarity between string vectors which represent texts as the reference table.

$$\begin{array}{c} \text{Word 1} \quad \text{Word 2} \quad \dots \quad \text{Word N} \\ \text{Word 1} \left[\begin{array}{cccc} S_{11} & S_{12} & \dots & S_{1N} \\ \text{Word 2} & S_{21} & S_{22} & \dots & S_{2N} \\ \dots & \dots & \dots & \dots & \dots \\ \text{Word N} & S_{N1} & S_{N2} & \dots & S_{NN} \end{array} \right] \end{array} \quad s_{ij} = \frac{2 \times \#(word_i, word_j)}{\#(word_i) + \#(word_j)}$$

Figure 2. Semantic Similarity Matrix

Let us mention the process of computing the similarity between string vectors as a semantic similarity between texts. The two string vectors which represent texts are notated by $\mathbf{str}_1 = [t_{11} \ t_{12} \ \dots \ t_{1d}]$ and $\mathbf{str}_2 = [t_{21} \ t_{22} \ \dots \ t_{2d}]$. The similarity between two string vectors is computed by equation (4),

$$sim(\mathbf{str}_1, \mathbf{str}_2) = \frac{1}{d} \sum_{i=1}^d sim(t_{1i}, t_{2i}). \quad (4)$$

The similarity between elements, $sim(t_{i1}, t_{2i})$, is taken from the similarity which was mentioned above. The value which is generated by equation (4) is always given as a normalized value between zero and one as shown in equation (5),

$$0 \leq sim(\mathbf{str}_1, \mathbf{str}_2) \leq 1. \quad (5)$$

Let us make some remarks on the encoding process and the computation of the similarity between texts. A text is encoded into a string vector by the word-text association, the feature definition, and the feature value assignment. The similarity matrix is defined as the basis for computing the similarity between two string vectors. The similarity between two string vectors is computed by equation (4). In the future research, we consider representing a text into multiple string vectors.

B. String Vector based KNN Algorithm

This section is concerned with the string vector based KNN algorithm. It was initially proposed as the approach to the text summarization by Jo in 2018 [6]. The similarity metric between string vectors which was described in the previous section is used for computing the similarity between a novice string vector and a training example. The labels of its nearest neighbors are voted with their identical weights for deciding the label of a novice input. This section is intended to describe the initial version of KNN algorithm from which its variants are derived.

The process of computing the similarity between a novice item and a training example is illustrated in Figure 00. The labeled words which are gathered as samples are encoded into string vectors, and they are notated by a set, $Tr = \{\mathbf{str}_1, \mathbf{str}_2, \dots, \mathbf{str}_N\}$. A novice word is encoded into a string vector notated by $\mathbf{str}$. Its similarities with the string vectors which represent the sample words are computed by equation (4), as $sim(\mathbf{str}, \mathbf{str}_1), sim(\mathbf{str}, \mathbf{str}_2), \dots, sim(\mathbf{str}, \mathbf{str}_N)$. Some training examples which are most similar as the novice input are selected as its nearest neighbors.

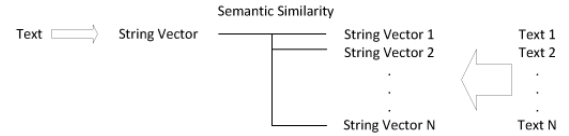


Figure 3. Similarities of Novice Input with Training Examples

In Figure 4, the process of selecting nearest neighbors by the ranking selection is illustrated. The training examples are sorted by the descending order of their similarities, after computing their similarities with a novice example. The training examples with their higher similarities with a novice example are selected as its nearest neighbors, as expressed in equation (6),

$$Nr = Select_k(Tr, \mathbf{str}), Nr \subseteq Tr \quad (6)$$

The set of nearest neighbors, Nr , is composed with elements as expressed in equation (7),

$$Nr = \{\mathbf{str}_1^{near}, \mathbf{str}_2^{near}, \dots, \mathbf{str}_k^{near}\} \quad (7)$$

The elements in the set, Nr , are used for voting their labels in the next step.

The process of voting the labels of nearest neighbors for decoding the label of a novice word is illustrated in Figure 5. The predefined categories are notated by $c_1, c_2, \dots, c_m$, and the label of a nearest neighbor is notated by $c_j = label(\mathbf{str}_i^{near})$. The number of nearest neighbors which belong to the category, c_j , is notated by $Count(Nr, c_j)$. The label of the novice item, $\mathbf{str}$, is decided by equation (8),

$$label(\mathbf{str}) = \arg\max_{j=1}^m Count(Nr, c_j) \quad (8)$$

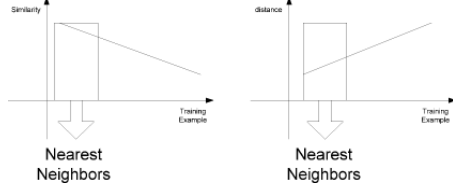


Figure 4. Selection of Most Similar or Least Distant Training Examples as Nearest Neighbors

The process of deciding the label of a novice item by equation (8), is called voting.



Figure 5. Voting of Labels of Nearest Neighbors for Deciding Label of Novice Input

Let us make some remarks on the KNN algorithm. It computes the similarities of a novice item with the training examples. The training examples with their highest similarities with the novice item are selected as its nearest neighbors. The label of the novice item is decided by voting the labels of its nearest neighbors. The ranking selection is adopted for selecting the nearest neighbors from training examples, in this version.

C. System Architecture

This section is concerned with the text segmentation system which adopts the string vector based KNN algorithm. In Section III-B, we described the proposed version of KNN algorithm as the approach to the text segmentation. It is viewed as a binary classification which classifies a paragraph pair into boundary or continuance. This section is intended to describe the text segmentation system with respect to its function and its architecture.

The process of gathering sample paragraph pairs and classifying a novice one, is illustrated in Figure 6. Because even a same paragraph pair may be classified differently depending on the domain, the task is called domain dependent classification. Sample paragraph pairs are gathered domain by domain, and each pair is labeled with boundary or continuance. The paragraph pairs are taken from texts which are tagged with their same domain, each paragraph pair is classified into boundary or continuance. Sample paragraph pairs which are labeled with one of the two categories are encoded into string vectors in each domain.

The entire architecture of the proposed text segmentation system is illustrated in Figure 7. A text is given as the input, and adjacent paragraph pairs are extracted by partitioning the text into paragraphs and slide the two sized window on them. The sample paragraph pairs in the boundary group and the

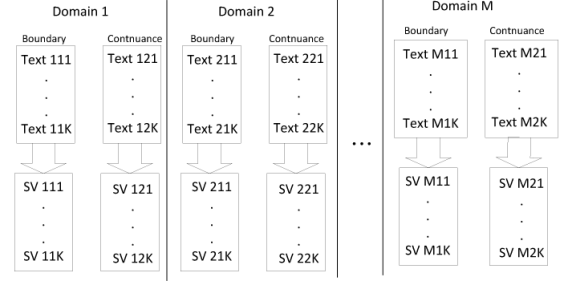


Figure 6. Preparation of Training Examples

continuance group and ones which extracted from the text are mapped into string vectors in the encoding module. The paragraph pairs from the text are classified into one of the two categories in the similarity computation module and the voting module. A boundary is put between paragraphs in each pair which is classified into boundary in the text, and it is partitioned into subtexts by the boundary.

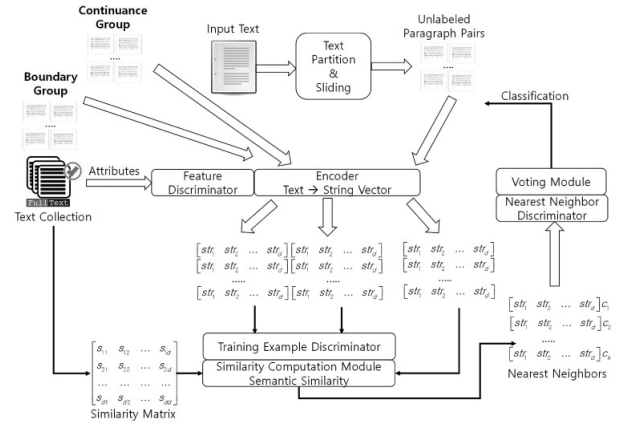


Figure 7. System Architecture

The execution flow of the text segmentation system is illustrated in Figure 8. Texts with same domain are initially collected, adjacent paragraph pairs are extracted from them, and the paragraph pairs are labeled with boundary or continuance. From an input text, adjacent paragraph pairs are extracted, and are encoded into string vectors, together with the sample paragraph pairs. The adjacent paragraph pairs are classified into boundary or continuance, and there are two groups of paragraph pairs: boundary group and continuance group. The boundary is marked between paragraphs in each pair in the boundary group, and text is partitioned by the marked boundaries into subtexts as the final output of this system.

Let us make some remarks on the proposed system which is illustrated in Figure 7 as its architecture. The text segmentation is defined as the binary classification where each adjacent paragraph pair is classified into boundary or continuance. The paragraph pairs are encoded into string

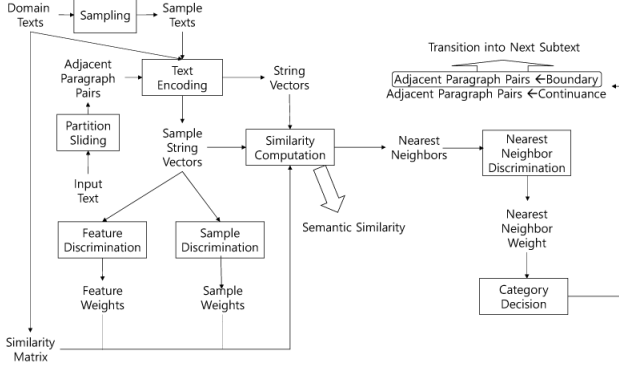


Figure 8. System Flow

vectors instead of numerical vectors, and they are classified into directly. The input text is partitioned into subtexts by the boundary which is put between paragraphs which are classified into boundary. The subtexts as the semantic division of the input text are treated as independent texts.

D. Connection with Text Clustering

This section is concerned with the connection of the text segmentation with the text clustering. In the previous section, we mentioned the text segmentation as an instance of domain dependent classification. The domains are automatically constructed from the text collection by clustering texts. The AHC algorithm which is proposed in [8] is used for implementing the text clustering. This section is intended to describe the text segmentation system which is connected with the text clustering for automating the construction of domains.

The architecture of the text clustering system which was proposed in [8] is illustrated in Figure 9. The texts in the group are encoded into string vectors, and the AHC algorithm which is proposed in [8] is adopted as the approach. It starts with singletons as many as items, computes the similarities of all possible cluster pairs, and merge the cluster pair with its highest similarity into one cluster. The two steps are iterated until the number of clusters reach the desired number. We may consider replacing the AHC algorithm by its variants for improving the speed and the performance.

The connection of the text segmentation with the text clustering is illustrated in Figure 10. The M domains are defined by clustering texts in a group, initially and automatically. A novice text is given as the input, and its domain, domain D, is decided by its similarities with domains. A classifier which corresponds to domain D is nominated and classify each adjacent paragraph pair in a novice text into boundary or continuance. The M index optimization systems are viewed as independent ones, each of which classifies each paragraph pair into one of the two categories.

The execution flow of text segmentation which is connected with the text clustering is illustrated in Figure 11.

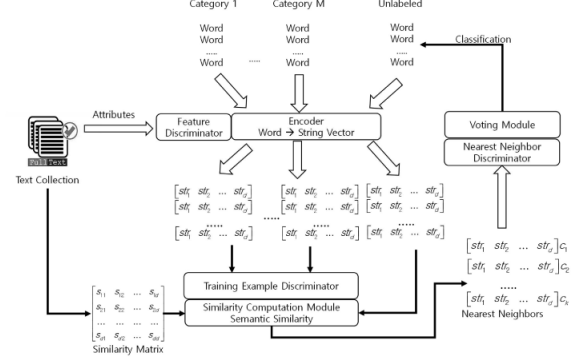


Figure 9. Text Clustering System

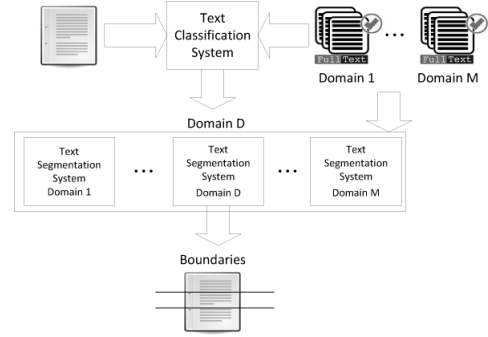


Figure 10. Text Segmentation System connected with Text Clustering

Unlabeled texts are clustered into subgroups, each of which is a domain. Sample paragraph pairs which are labeled with boundary or continuance are generated from each domain. These sample paragraph pairs are provided for training the classifier in each text segmentation system. Each classifier is used for judging whether the topic change happens between paragraphs in each pair, or not.

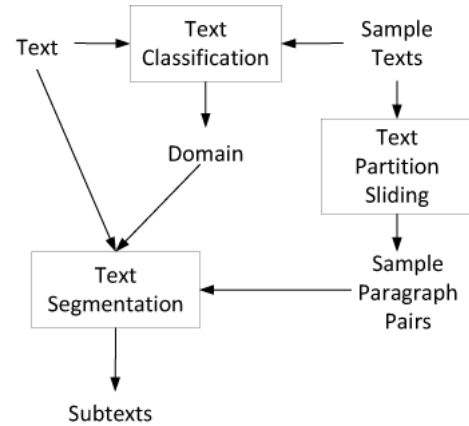


Figure 11. System Flow of Text Segmentation connected with Text Clustering

Let us make some remarks on the connection of the text

segmentation with the text clustering. It is the process of segmenting a text group into subgroups based on their content-based similarities. The role of text clustering is to define the domains initially in the architecture of connecting the text segmentation with the text clustering. In each domain, sample paragraph pairs are generated, and its corresponding classifier is trained with them. In the future research, we consider using the text segmentation for clustering texts in the opposite direction.

IV. CONCLUSION

Let us mention the significances of this research as the conclusion. We modify the KNN algorithm into the string vector based version. We design the text segmentation system by adopting the proposed KNN algorithm. We connect the system with the text clustering system for automating the domain decision. In the next research, we will implement the text segmentation system as a real system.

Let us mention some remaining tasks as further discussions on this research. More advanced machine learning algorithms and deep learning algorithms are modified into string vector-based versions. We need to define and characterize more semantic operations on string vectors. The text segmentation system which relates to the text clustering may be implemented as a real program or a library. The text classification and the text clustering relate to the text segmentation for improving their performances.

REFERENCES

- [1] T. Jo, "NTC (Neural Text Categorizer): Neural Network for Text Categorization", 83-96, International Journal of Information Studies, 2, 2, 2010.
- [2] T. Jo, "NTSO (Neural Text Self Organizer): A New Neural Network for Text Clustering", 31-43, Journal of Network Technology, 1, 1, 2010.
- [3] K. Abainia, S. Ouamour, and H. Sayoud. "Neural Text Categorizer for topic identification of noisy Arabic Texts." 1-8, The Proceedings of IEEE/ACS 12th International Conference of Computer Systems and Applications, 2015.
- [4] T. Jo, "String Vector based AHC for Text Clustering", 673-677, The Proceedings of 18th International Conference on Advanced Communication Technology, 2017.
- [5] T. Jo, "Long Text Segmentation by String Vector based KNN", 805-809, The Proceedings of 18th International Conference on Advanced Communication Technology, 2017.
- [6] T. Jo, "Automatic Text Summarization using String Vector based K Nearest Neighbor", 6005-6016, Journal of Intelligent and Fuzzy Systems, 35, 2018.
- [7] P. P. Lakshmi and D. R. Rao, "Text classification using artificial neural networks", 603-606, International Journal of Engineering & Technology 7, 1, 2018.
- [8] T. Jo, "Semantic String Operation for Specializing AHC Algorithm for Text Clustering", 1083-1100, Annals of Mathematics and Artificial Intelligence, 2019.
- [9] T. Jo, "Introduction of String Vectors to AHC Algorithm for Clustering News Articles", 150-153, The Proceedings of 21st International Conference on Artificial Intelligence, 2020.
- [10] T. Jo, "Application of String Vector based K Nearest Neighbor to Content based Segmentation of each News Article", 58-64, The Proceedings of 2nd International Conference on Advanced Engineering and ICT-Convergence, 2019.
- [11] T. Jo, "Text Segmentation based on Contents using String Vector based Version of K Nearest Neighbor", in Preparation, 2023.